Carnegie Mellon
Software Engineering Institute

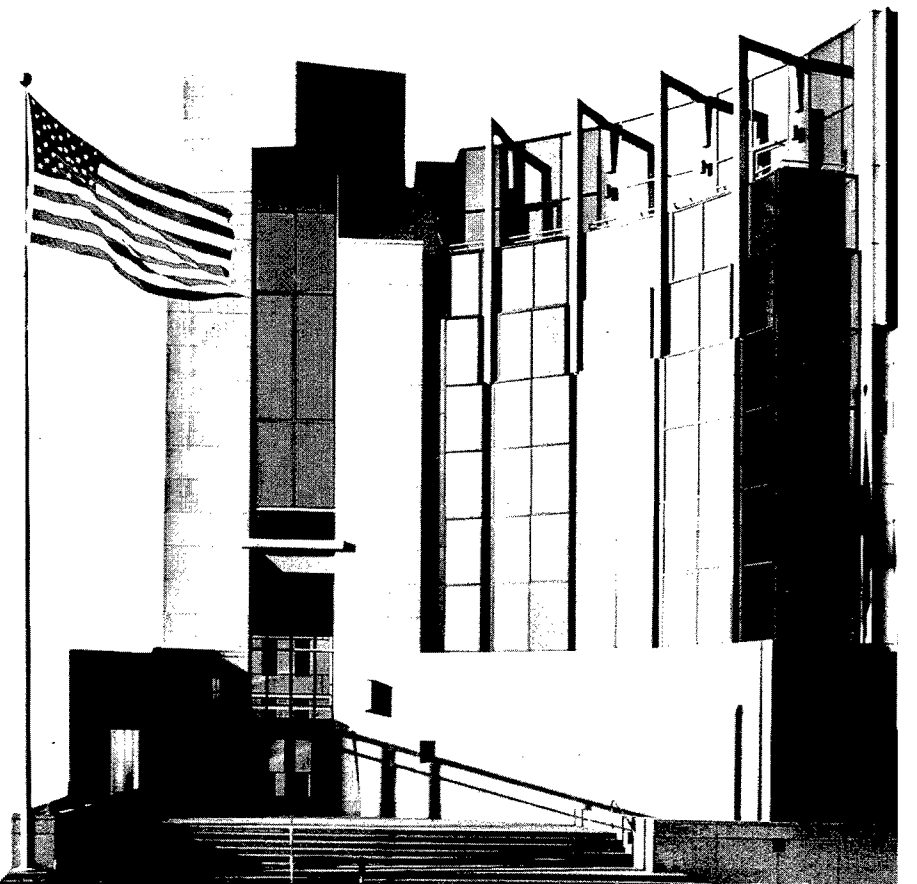
Topics in Interoperability: System-of-Systems Evolution

David Carney
David Fisher
Patrick Place

March 2005

Integration of Software-Intensive Systems Initiative

Technical Note
CMU/SEI-2005-TN-002



Topics in Interoperability: System-of-Systems Evolution

David Carney
David Fisher
Patrick Place

March 2005

Integration of Software-Intensive Systems Initiative

Technical Note
CMU/SEI-2005-TN-002

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Contents

Abstract v

1 Introduction 1
- 1.1 Interoperation as a Relationship 1
- 1.2 Depicting Interoperation 2
- 1.3 Boundaries of Systems and Systems of Systems 3
- 1.4 Relationships Implemented by Systems 5

2 Notions of Software Evolution 6
- 2.1 Existing Research on Software Evolution 6
- 2.2 Drivers of Software Evolution 7
- 2.3 System Characteristics that Affect Evolution 8

3 Evolution in the Context of Systems of Systems 9
- 3.1 Motivation for Evolution of a System-of-Systems 9
- 3.2 Locality of the Evolutionary Process 10
- 3.3 Outcome of Evolution 10

4 Properties of Evolution that Affect Interoperability 11

References 13

List of Figures

Figure 1: Systems and Relationships 3

Figure 2: Systems and Systems of Systems 4

Figure 3: Evolution and Interoperability 11

Abstract

This report examines how interoperable systems of systems evolve. It first considers several ways in which interoperability can be defined and then examines the notion of software evolution itself. Next, it considers how evolution occurs in interoperable systems of systems by discussing issues such as the motivation for and outcome of evolution. Finally, it proposes several properties of evolution that directly affect interoperability—in particular, how interoperability can be maintained as the individual systems evolve.

This report is the first in a series of reports on interoperability. This series will consider the various properties and attributes of interoperability in an effort to determine how to measure the ability of a system to interoperate with other systems; predict the resources needed for successful interoperation; and discover techniques useful to achieving interoperability.

1 Introduction

The topic of this paper is the evolution of interoperable systems of systems. We begin with a brief discussion of interoperability in general that sets out several key concepts that underlie the remainder of the paper.

1.1 Interoperation as a Relationship

The term *interoperability* has many definitions; a reasonable one is

> The ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to a shared operational semantics [Brownsword 04].

For the purposes of the discussions that follow, we extend the above definition by adding the notions of purpose (the goal for the interoperation) and context (the environment in which the entities exist). This leads to a definition of *interoperability* as

> The ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to a shared operational semantics in order to achieve a specified purpose in a given context.

The essence of interoperation is that it is a **relationship** between **systems**, where systems are the entities in the above definition. While our focus will be on computer-based systems, the definition extends to beyond the world of mechanical systems to organizational and other contexts. To interoperate one system must provide a service[1] that is used by another. This cannot be achieved without, at a minimum, communication from the provider to the consumer of the service. Our focus is the relationship and not the manner of communication.

Interoperability relationships **necessarily** involve communication. Just as in the physical world a relationship of proximity may not involve interoperability (e.g., the table is **close to** the chair), a proximity relationship in the software domain may not involve interoperation. For instance, the mere fact that two software systems are both installed on a single machine does not imply that they are interoperable (though they might, of course, be interoperable by some other relationship).

For the purposes of this report, we do not take a position with regard to the atomicity of interoperability relationships. We might define the relationship between two entities as being

[1] While it seems obvious, it must be stated that provision of service includes the provision of data.

a single relation that contains multiple communications or as a collection of relations where each relation is a single communication. For now, we allow interoperability relations to be split or combined as broadly as seems useful within the constraints that each interoperability relationship involves (1) multiple[2] systems (2) service provision and use, and (3) essential inter-system communication.

An interoperability relationship need not be transitive, commutative, or reflexive. In the first case, A can provide a service to B, and B can provide a service to C without A providing a service to C. In the second case, A can provide a service to B without B providing that (or for that matter any other) service to A. In the third case, A need not be a consumer of its own services. Note, however, that an interoperability relationship *may* be transitive, commutative, or reflexive. For example, in some systems A may provide some service to B and B may provide the same service to A. In the composition of systems we expect to see many different kinds of interoperability.

Software interoperability relationships can be of many possible kinds and degree, and can be brought about by many different implementation mechanisms. For instance, we can describe some relationships between software systems as "tightly coupled," and other relationships as "loose." We can implement an interoperability relationship by means of capabilities entirely within the communicating entities (e.g., an agreement to share a common protocol), or the relationship can be implemented by some other software entity (e.g., a request broker that relays messages between systems).

1.2 Depicting Interoperation

Throughout this paper we use the simplest graphical means to represent systems and relationships. We depict systems simply as ellipses (or circles[3]), and a relationship between systems as a straight line that connects the ellipses. In the figure below, there are three systems (A, B, and C) that have interoperability relationships of some kind.

[2] The notion of relation is easily extended to more than two systems. Such extension does not alter the fundamental concepts of this report.

[3] This is formally consistent: a circle is an instance of an ellipse whose major and minor axes are the same.

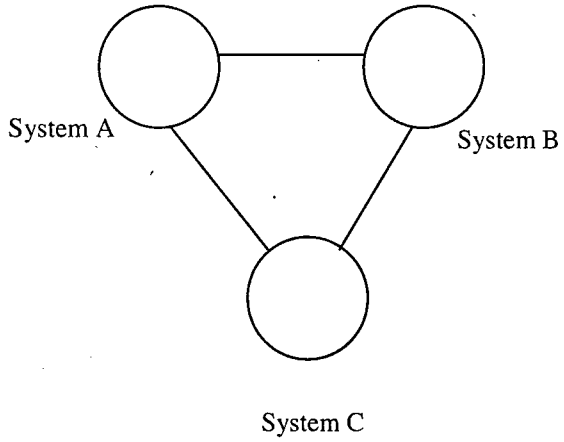


Figure 1: Systems and Relationships

As we will more fully discuss in the next section, we do not attempt graphically to convey any further information. Thus, the three lines above may represent the same relationship or may be quite different, and the "closeness" or "tightness" of the relationships may be similar or different for all of the lines. Similarly, the systems represented in the above diagram may be large or small, trivial or complex; and any or all of the circles may even themselves be comprised of smaller systems.

1.3 Boundaries of Systems and Systems of Systems

Almost every discussion of interoperability is plagued by one annoying reality: any construct that we label "a system" may in fact be composed of several constituent systems, and this may recursively be true at several levels. In other words, anything that at one level we can call a "system" may actually internally be a "system of systems," and any "system of systems" may itself be part of some larger "system of {systems of systems}," and so forth.

To illustrate, we imagine some hypothetical data systems that interoperate in some manner. These data systems could all be elements (e.g., communication or navigation) of a military aircraft's avionics system, which together with many other systems (weapons system, mission management system) compose the total aircraft, which itself can be viewed as a single system. To continue to even higher levels, the aircraft is an element in a larger system of systems, since it interoperates with other aircraft and other military units in combat. The process can continue recursively through ever larger systems of systems of systems of systems.

To facilitate our discussion of interoperability, we need to define some level of immediate interest. To do so we choose one of these many levels as that of "the system" and the next higher level that of the "system of systems." The level we choose is, in a sense, artificial,

since it is only one vantage point within the potentially large scope of this recursive sequence. But it is useful to focus discussion and analysis.

Thus, if our concern at the moment is with issues related to low-level data, semantics of data, and so forth, we could choose the data systems noted above and their interoperability relationships as our level of interest.

We illustrate this as follows:

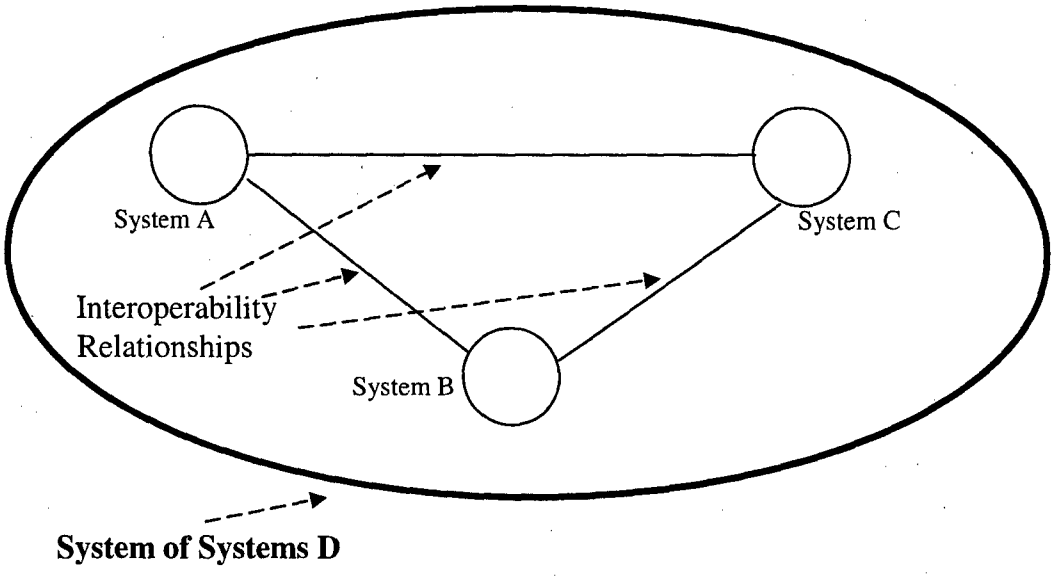


Figure 2: Systems and Systems of Systems

Graphically, therefore, we let the three smaller circles above represent the individual data systems in the hypothetical example described earlier. Each is related to two others by some interoperability relationship. The three together as a related unit, that is, the "system of systems" is depicted by the large darker oval; this would be the hypothetical avionics system. The smaller circles may themselves each comprise several systems, and the large oval may itself be a single system in some larger context. We temporarily ignore those possibilities and focus only on the interrelationships between A, B, and C that bring about D. By choosing this particular vantage point, we are able to consider the precise nature of the three constituent systems, their interrelationships, and the principles by which the system of systems (D) is brought about.

At some later time, if our concern lies in some other sphere (e.g., real-time factors relating to the avionics and weapons systems), then our level of discourse could well be the interoperability relationships at that level, and so on.

1.4 Relationships Implemented by Systems

A further facilitating device is that we use a common vocabulary regardless of the mechanism by which a relationship is implemented. For example, we can imagine two systems (A and B) whose relationship is such that they must communicate data back and forth. Let us further suppose that the relationship is implemented by some complex communication system. Since that communication system is, by definition, a system in its own right, it is easy to see that discussion of such a collection may easily be complicated by two different opinions. One opinion sees a system of systems of three entities (A, B, and the communication system). The other opinion sees a system of systems of only two (i.e., by disregarding that the communication system is a *system*, and viewing it only as implementing the relationship between A and B).

We argue that either view is possible, depending on what issues are of immediate interest and what questions are being asked. For instance, we may be interested in the semantics of shared data between A and B, and are unconcerned with the manner in which the data is communicated. In that case, we can rightly consider the communication system simply as the mechanism that implements the A-B relationship. On the other hand, we may be concerned with the specific details of how system A locates system B, with the significance of timing constraints and other such questions. We might well then consider that the relationships between system A, system B, and the communication system are all interoperability relationships in their own right.

We now turn to the topic of software evolution and the ways in which interoperating software systems evolve. Section 2 provides a summary of some major research in software evolution. Section 3 considers evolution in the context of systems of systems. Section 4 sets forth some properties of interoperability. Section 5 provides a brief summary of the report.

2 Notions of Software Evolution

Much of the research in software evolution has understandably dealt with evolution of individual systems, which is somewhat removed from our concern with evolution of systems of systems. However, it is useful to consider briefly some general ideas about evolution of independent software systems, since these ideas are no less significant in the context of evolving systems of systems. This chapter will therefore concentrate on evolution in the context of single systems, and the following chapter will focus on evolution in the context of systems of systems.

2.1 Existing Research on Software Evolution

Of the numerous definitions of evolution, we posit the following as reasonable:

> Evolution is any change in the quality, functionality, or implementation of the services offered by a system.

Many other definitions can easily be found, and most of them are more or less a paraphrase of this statement. Some definitions (and hence the attendant research) in the field is concerned with software evolution in terms of its processes:

> Software evolution is the set of activities, both technical and managerial, that ensures that software continues to meet organizational and business objectives in a cost effective way [RISE 99].

Other research focuses more on the pragmatic, searching for tools and techniques:

> Software systems need to evolve continuously to cope with the ever-changing software requirements. Today, this is more than ever the case. Nevertheless, existing tools designed to provide support for evolution issues are far from ideal. They are typically developed in an ad-hoc fashion, making them not generally applicable, not scalable, or difficult to integrate with other tools. The goal of this research network is to come to a consistent set of formally founded techniques and associated tools to support software developers with the common problems they encounter when developing large and complex software systems [Scientific 01].

Some view evolution as largely synonymous with "maintenance" (e.g., seen in such phrases as "Software Evolution, a.k.a Maintenance"). Ian Sommerville, in his "Software Engineering" 6th ed., takes a more nuanced position [Sommerville 00]. For Sommerville,

evolution is the broader concept, of which maintenance is one possible strategy (others being architectural transformation and re-engineering).

Of considerable importance are the eight "laws of software evolution," originally described by Lehman several decades ago, and now the foundation of the Feedback, Evolution, and Software Technology (FEAST) project [Lehman 00]. Lehman states these rules in terms of systems that are actively used and embedded in a real world domain. Such systems are judged by the results that they deliver and by user satisfaction; they are *not* judged in terms of how well they meet a well defined set of requirements.

For our purposes, the most relevant of Lehman's laws are the first and sixth laws:

1. Systems must be continually adapted else they become progressively less satisfactory.
6. The functional content of systems must be continually increased to maintain user satisfaction over their lifetime.

2.2 Drivers of Software Evolution

Lehman's first law suggests that for a real-world system, evolution is unavoidable. The force of this law has grown since its first formulation and is increasingly apparent today, when we witness near-breakneck rates of change in the software domain. Since any modern software system has numerous connections with its environment, including other systems with which it communicates or depends, no system can survive such rapid environmental change without evolving correspondingly.

We suggest three broad categories of factors that drive evolution. Evolution of a system must occur in response to

- changing needs of its users
- the improved effectiveness of adversaries such as hackers or intruders (which reflects the present tendencies in society and technology)[4]
- changing technology

The "changing needs of users" can be of various kinds. Chief among them are those related to marketplace forces: the kinds and numbers of users may change; user expectations may grow in response to competitive services; or user expectations may grow in response to advances in technology. In addition, however, there are needs invisible to users, but that may still drive system evolution. For instance, a system's creators may better understand the need to introduce a new, low-level infrastructure technology, and may evolve the system accordingly.

There is plentiful evidence that "changing technology" can drive software evolution. (Note that the technology in question may or may not be specifically in the software domain.)

[4] The notion that adversaries become more effective may be thought of as a special case of changing needs of a system's users. It is generally assumed that one of the user needs, even if unstated, is that a system will survive adversarial efforts, regardless of the nature of those efforts.

Whatever their domain, however, the changes that drive software evolution may sometimes occur to improve quality or cost of services, or may simply satisfy previously unmet needs. Another driver in this category is the need to replace end-of-life technologies and to keep pace with competitive services.

Finally, evolution in the face of improved effectiveness of adversaries is fast becoming a dominant challenge for all software practitioners. We consider "adversaries" in the widest sense, from creators of viruses to hackers launching "denial of service" attacks to the ubiquitous spammers: all are in some manner adversarial forces that drive evolution of software systems from their present vulnerable state. And since these adversaries themselves evolve, our systems must keep pace. These adversaries gain greater understanding of current vulnerabilities and exploit advances in technology. As software literacy continues to spread throughout different cultures and nations, the kinds and numbers of adversaries themselves change and evolve. Finally, economic drivers can be as strong as technological drivers: as some software systems become more and more successful, this in itself magnifies their attractiveness and value as a target.

2.3 System Characteristics that Affect Evolution

To fully understand the evolution of software systems, we need to understand that there are certain characteristics of a system that might promote evolution, and other characteristics that might hinder it. (Note that in all cases, we are concerned only with systems that operate in the physical world and are subject to marketplace and other comparable forces; these are the same systems addressed by Lehman's laws.)

There are many such characteristics, more than we can list here. However, for many of these characteristics, there appears to be a natural tension between stability and flexibility, and between promoting and hindering evolution. In other words, many characteristics of a system that are desirable from the viewpoint of having a static, stable system are precisely those that hinder evolution, and vice-versa. Thus,

- the more interconnections in a system, the more that system is stable and resistant to change
- the more interconnections in a system, the more that system is brittle and easy to break during change

However, stability is only one characteristic that is antagonistic to evolution. For example,

- the more optimized a system, the higher its performance
- the more optimized a system, the lower its ability to evolve

In the following sections, we shall consider these and other characteristics of systems and system evolution in the context of interoperability, and will suggest a number of system-of-system characteristics that either promote or hinder evolution.

3 Evolution in the Context of Systems of Systems

When systems are related by some interoperability relationship, all of the evolutionary issues described in the previous chapter remain true for each system. In addition, however, another dimension appears, namely, the evolution of the system of systems, which includes, but is also distinct from, the individual systems' evolution. It is the interoperability relationship on which we concentrate, since its evolution is of primary significance to us.

We examine the evolution of systems of systems by considering the following questions:

- *Why* does the system of systems evolve?
- *Which parts* of it evolve?
- *What changes* are brought about (i.e., what is the difference between the "before" and "after" states)?

The first question concerns the motivation for evolution, the second concerns the locality of the evolutionary process, and the third concerns its outcome. These questions are not truly independent; they simply reflect slightly different ways to consider system-of-systems evolution. We discuss each in turn.

3.1 Motivation for Evolution of a System-of-Systems

In general, the motivation for evolution of a system of systems includes all of the possible motivations as for individual systems: Lehman's laws, particularly the first (i.e., that systems must continually evolve) are no less true when multiple systems are interoperating.

Thus, the individual systems (the ellipses in our simple diagrams) will periodically change and evolve for various reasons. For instance, new releases of COTS components may occur. Or the number of systems within the system of systems may vary as new systems are introduced and old systems are retired. As such events occur, the interoperability relationships necessarily evolve to accommodate them. Thus, this kind of evolution is termed *preservative.* The goal is preservation, but the system of systems is otherwise unchanged in terms of its functioning, the mission it fulfils, and its general shape and architecture. Note that the 'preservation' on which we focus is that of the *interoperability relationship* itself.

By contrast, some evolution is *adaptive*. Thus, there may arise some new or different mission that the system of systems must fulfill, which requires new, different functionality, or addition of new or different relationships between existing systems, or addition of new systems. We also include those evolutions that occur when relationships or component systems are removed, since these are also forms of adaptive evolution. Once again, the entity of interest that is adapting is the *interoperability relationship.*

3.2 Locality of the Evolutionary Process

Assuming that some set of systems is interoperable, we posit that there are different locales that could be the primary location of any evolutionary process. In other words, evolution might be principally an event for a single system in itself, or principally in one or more individual relationships, or could be equally spread throughout the entire aggregate system of systems. Note that we are only pointing out a primary locale of some evolutionary event; we believe that evolution of any element in a system of systems will likely necessitate at least some evolutionary activity in other elements.

3.3 Outcome of Evolution

There are three potential outcomes for an evolutionary event:

- The evolution produces a new system of systems that largely resembles the original. This occurs through normal modernization practices, COTS updates, keeping pace with technology, and so forth.
- The evolution produces a significant expansion from the original, generally in numbers of systems and relationships. This may occur through change of mission, when new functionality becomes necessary. This may also occur as a result of rearchitecting, as when two large, complex interoperating systems are broken into several smaller interoperating systems. While such rearchitecting does not necessarily involve significant expansion of functionality, it certainly expands the number of relationships in the overall system.
- The evolution produces a significant contraction from the original. This is reverse of the previous type, and could occur for comparable reasons. Thus, a changed mission could provoke a reduction in functionality, or a decision could be made to collapse several small independent systems into a smaller number of large systems.

These are abstract distinctions, and real-world situations will seldom have a precise fit with them. But they are useful in that they provide a framework to reason about how interoperability relationships evolve, what promotes their successful evolution, and what hinders it.

4 Properties of Evolution that Affect Interoperability

There are many properties of either evolution or interoperability that could be considered separately. For our purposes, it is only properties of both that concern us. Figure 3 denotes this: the properties of evolution are the left circle and properties of interoperability are the right circle. Our concern is only those properties in the circles' intersection that apply to both.[5]

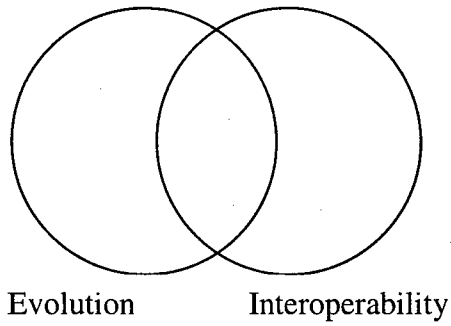


Figure 3: Evolution and Interoperability

Given that individual systems must evolve, maintenance of their interoperability relationships is essential to fulfillment of the overall purpose or mission. Below we list eight properties of evolution and interoperability that affect how that maintenance will be achieved. We state these properties in terms of their effect (i.e., either to facilitate or to hinder) the maintenance of interoperability as a system of systems evolves.[6]

1. **Relative stability of the components.** It is likely that different systems will evolve at different rates. However, if individual systems are relatively stable, that is to say that there is some synchronization of changes among the systems, then there is an increased likelihood that interoperability will be maintained.
2. **Existence of agreements regarding the evolution between systems affected by the changes.** The more that owners of systems can make local agreements with respect to change, the more likely local interoperability relationships will be preserved. This tends toward, but does not guarantee, preservation of global interoperability. Where there are

[5] By symmetry, if the goal were to define evolution, then the intersection defines those properties of interoperability that affect evolution.

[6] In this list of properties, 'local' refers to activity either within an individual system or to a single relationship. *Global* refers beyond a single system or relationships, most often to the system of systems as a whole.

few or no local agreements concerning change in any one system, every other system will be forced to react to change rather than harmonize with it.

3. **The number of interoperability relationships in the system of systems.** The greater the aggregate number of relationships, the harder it is for any one system to evolve without requiring evolution in many other systems. A large number of relationships requires greater coordination than when there are fewer relationships.

4. **The number and complexity of interoperations affected by the change.** Given that any system may be involved in more than one interoperability relationship, it follows that the greater the number of relationships affected by a change, the harder it will be to maintain global interoperability.

5. **Coordination of communication among systems.** As systems evolve there is a reasonable probability that the rates at which they interoperate will vary. However, the more closely coordinated the communication rates in any local interoperability relationship, the more effective the relationship will be. Thus, coordinating rates of communication is an aspect of maintaining interoperation.

6. **Commonality of purpose among component systems.** Our definition of interoperability used the notion that systems interoperate in order to achieve some purpose. Hence, the more closely each system is aligned with that purpose, the more willing the system's owners will be to accommodate changes. For example, if a service provided by system A is peripheral to the purpose of system B, then changes in A that decrease local interoperability between A and B will tend to be ignored, and B will look for some other provider of the service. Note that this is true regardless of the diversity of these components.

7. **The ability to assess trust in the face of change.** A key aspect of interoperability is the need for one system to establish trust[7] in another. Indeed, not only must trust be established but also must be constantly re-evaluated. Such re-evaluation is particularly necessary with every evolution of a trusted system.

8. **Adaptability of components.** Given the notion that all systems are continually evolving and that the context for those systems is also evolving, it follows that each system must be continually adapting to its new context. For example, if a system is adaptable with respect to different communication rates, then it is likely that the interoperability relationships will be preserved even though the competition for communication resources changes.

[7] This is particularly true when communication is machine-to-machine. While trust is really a concept based on human interaction it is clear that some facsimile of trust must be developed for machine to machine interaction. As an example, one aspect of trust could be whether or not the data just received from some other system is the most recent valid instance of the data. More will be written about properties of trust in a future technical note.

References

URLs are valid as of the publication date of this document.

[Brownsword 04] Brownsword, L et. al. *Current Perspectives on Interoperability* (CMU/SEI-2004-TR-009). Pittsburgh, PA: Software Engineering Institute, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tr009.html

[Hearnden 04] Hearnden, D. "Software Evolution with the Model-Driven Architecture." http://www.itee.uq.edu.au/ ~hearnden /_deltaware/ConfirmationSeminar.ppt (2004).

[FEAST 01] Feedback, Evolution and Software Technology (FEAST) Project. London, UK: Imperial College, Department of Computing. http://www.doc.ic.ac.uk/~mml/feast2/index.html (2001).

[Lehman 00] Lehman, M. *Rules and Tools for Software Evolution Planning and Management.* London, UK: Department of Computing Imperial College, 2000. http://www.doc.ic.ac.uk/~mml/feast2/papers/pdf/611_2.pdf

[RISE 99] Research Institute in Software Evolution (RISE). Durham, UK: Dept of Computer Science, University of Durham. http://www.dur.ac.uk/CSM/ (1999).

[Scientific 01] Scientific Research Network, Foundations of Software Evolution, Flanders, Belgium. http://prog.vub.ac.be/FFSE/network.html (2001).

[Sommerville 00] Sommerville, I. *Software Engineering, 6th ed.* London, UK: Pearson Education, 1982, 2000. http://www.comp.lancs.ac.uk/computing/resources/IanS/SE6/index.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	March 2005	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Topics in Interoperability: System-of-systems Evolution	F19628-00-C-0003

6. AUTHOR(S)

David Carney, David Fisher, Patrick Place

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2005-TN-002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This report examines how interoperable systems of systems evolve. It first considers several ways in which interoperability can be defined and then examines the notion of software evolution itself. Next, it considers how evolution occurs in interoperable systems of systems by discussing issues such as the motivation for and outcome of evolution. Finally, it proposes several properties of evolution that directly affect interoperability—in particular, how interoperability can be maintained as the individual systems evolve.

This report is the first in a series of reports on interoperability. This series will consider the various properties and attributes of interoperability in an effort to determine how to measure the ability of a system to interoperate with other systems; predict the resources needed for successful interoperation; and discover techniques useful to achieving interoperability.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Interoperability, systems of systems, system evolution, interoperability measurement	23

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL